MASTER COPY KEEP THIS COPY FOR REPRODUCTION PURPOSES

REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 12, 1990	Technical

4. TITLE AND SUBTITLE

Computer Program for Mode Search in Partially-filled Waveguide

5. FUNDING NUMBERS

DAAL03-87-K-0088

6. AUTHOR(S)

T. E. van Deventer, P. B. Katehi

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

University of Michigan
Ann Arbor, MI 48109

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

U. S. Army Research Office
P. O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

ARO 23836.14-EL

11. SUPPLEMENTARY NOTES

The view, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

The FORTRAN program PFW.FTN (Partially-Filled Waveguide) was written to compute the complex propagation constant of the hybrid modes propagating in an inhomogeneously-filled waveguide with any number of lossy substrates and non-perfectly conducting walls.

14. SUBJECT TERMS

Computer Programs, Mode Search, Waveguides, Fortran

15. NUMBER OF PAGES
9

16. PRICE CODE

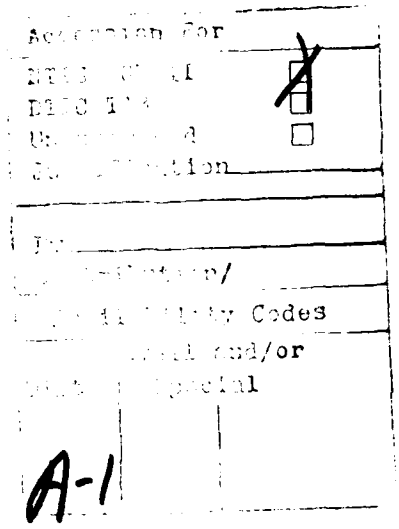
17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Computer Program for Mode Search in Partially-filled Waveguide

T.E. van Deventer, P.B. Katehi

Radiation Laboratory
Department of Electrical Engineering
and Computer Science
University of Michigan
Ann Arbor, MI 48109

March 12, 1990



This work was supported by the Army Research Office under contract project DAAL03-K-0088 (23836-EL).

Computer Program for Mode Search in Partially-filled Waveguide

T.E. van Deventer, P.B. Katehi

Radiation Laboratory
Department of Electrical Engineering and Computer Science
University of Michigan
Ann Arbor, MI

GOAL

The FORTRAN program PFW.FTN (Partially-Filled Waveguide) was written to compute the complex propagation constant of the hybrid modes propagating in an inhomogeneously-filled waveguide with any number of lossy substrates and non-perfectly conducting walls.

FORMULATION

The general version of the program solves for the hybrid TE and TM to x modes that propagate in structures as shown in Figure 1. To determine the possible propagation constants k_z, the following transcendental equations corresponding to these hybrid modes are solved for LSM (TM_x) modes as

$$\frac{k_{x1}}{\epsilon_1} tan k_{x1} h_1 = -\frac{k_{x2}}{\epsilon_2} tan k_{x2} h_2 \quad (1)$$

and for LSE (TE_x) modes

$$\frac{k_{x1}}{\mu_1} cot k_{x1} h_1 = -\frac{k_{x2}}{\mu_2} cot k_{x2} h_2 \quad (2)$$

with

$$k_{x1}^2 + (\frac{n\pi}{b})^2 + k_z^2 = \omega^2 \epsilon_1 \mu_1 \quad (3)$$

$$k_{x2}^2 + (\frac{n\pi}{b})^2 + k_z^2 = \omega^2 \epsilon_2 \mu_2. \quad (4)$$

The hybrid modes are denoted LSE_{mn} and LSM_{mn}. The transcendental equations have an infinite number of solutions for a given mode number n. The index m denotes the order of these solutions. For $b > a$, the dominant mode is the LSM_{01} mode ($m = 0$, $n = 1$) followed by $m = 1$, and so on. For $b < a$, the first mode is an LSE mode with $n = 0$. In this case the numbering on the index m starts at $m = 1$, then $m = 2$, and so on. The numbering is chosen to be consistent with that of the empty waveguide [1].

The equations (1), (2) are derived using the transverse resonance method, since the structure at resonance reduces to a transmission-line structure with propagation in the x-direction. In the case of perfectly conducting walls, the end transmission-lines are shortcircuited as shown in Figure 2. However finite conductivity of the metal housing has been implemented in the PFW.FTN where the top and bottom walls can have a conductivity σ, which in turns can be described by a load at the end of the transmission line. The program also accounts for substrate losses with the option of inputting the dielectric and magnetic loss tangents.

$$\epsilon_i = \epsilon_{ri}(1 - j \tan \delta_i) \quad (5)$$

$$\mu_i = \mu_{ri}(1 - j \tan \gamma_i). \quad (6)$$

This program allows to compute the complex propagation constant of any propagating mode in a rectangular waveguide structure, as a function of frequency. It can also calculate the cut-off frequency of the modes by choosing a frequency increment small enough around $k_z = 0$.

DESCRIPTION

The geometry of the structure is inputted through a user-friendly menu program. The output is written in a file and on the screen,

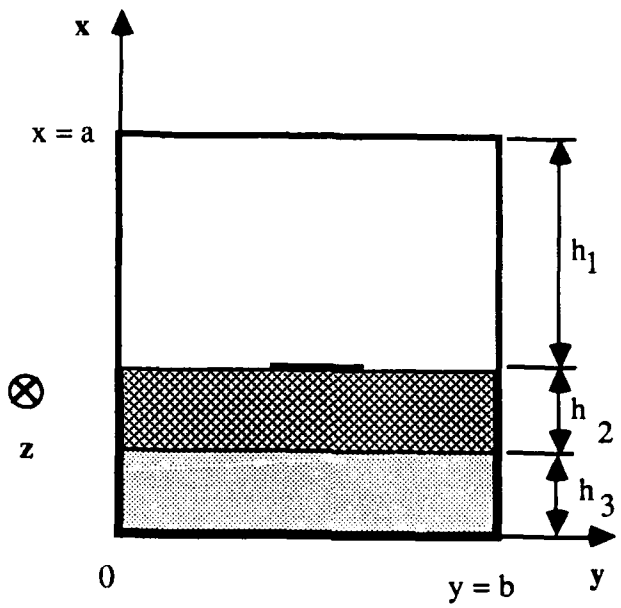


Figure 1: Partially-filled waveguide configuration

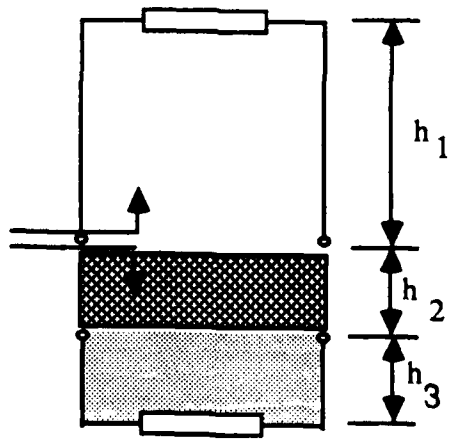


Figure 2: Transmission-line Analogy

and gives the propagation constant of the modes propagating within the frequency range of interest.

.1 menu

The menu is an interactive program that allows for on-screen creation of the input data file. Also changes can be made on the screen by entering the variable name to be changed (e.g. FSTR to change the starting frequency). The program will then prompt for a new value of the variable. Dimensions can be entered either in centimeters or inches. Conductor losses in the top and bottom walls may be included, in which case the conductivity of the walls should be given. The maximum mode number n refers to the discussion in the previous section. The output is written in a file called 'RESULT'.

.2 program

The PFW.FTN program calculates the roots of (1) and (2) for the lossless case. When losses are considered, the roots of the transcendental equations are found using Muller's method with deflation. The present program calls for the following subroutines of the *IMSL* library (November 1, 1984 release) [2]

- ZANLYT (find the zeros of a univariate complex function using Muller's method)
- UGETIO
- USPKD
- UERTST.

These subroutines have to be bound to the main program PFW.FTN for the program to run. If this library is not available, a lossless version of the program may be requested.

.3 limitations

The program PFW.FTN is a generalized code that can handle any number of dielectric layers. However, for programming purposes, the arrays have been dimensionned to a maximum of 20 layers.

VERIFICATION

Using the approach described in the previous section a computer program was developed to calculate the complex propagation constant. The validity of this program has been verified in the case of lossless substrates. Good agreement with Yamashita's paper [3] is shown, and the results of the program PFW.FTN given in the next section may be compared to Figure 3.

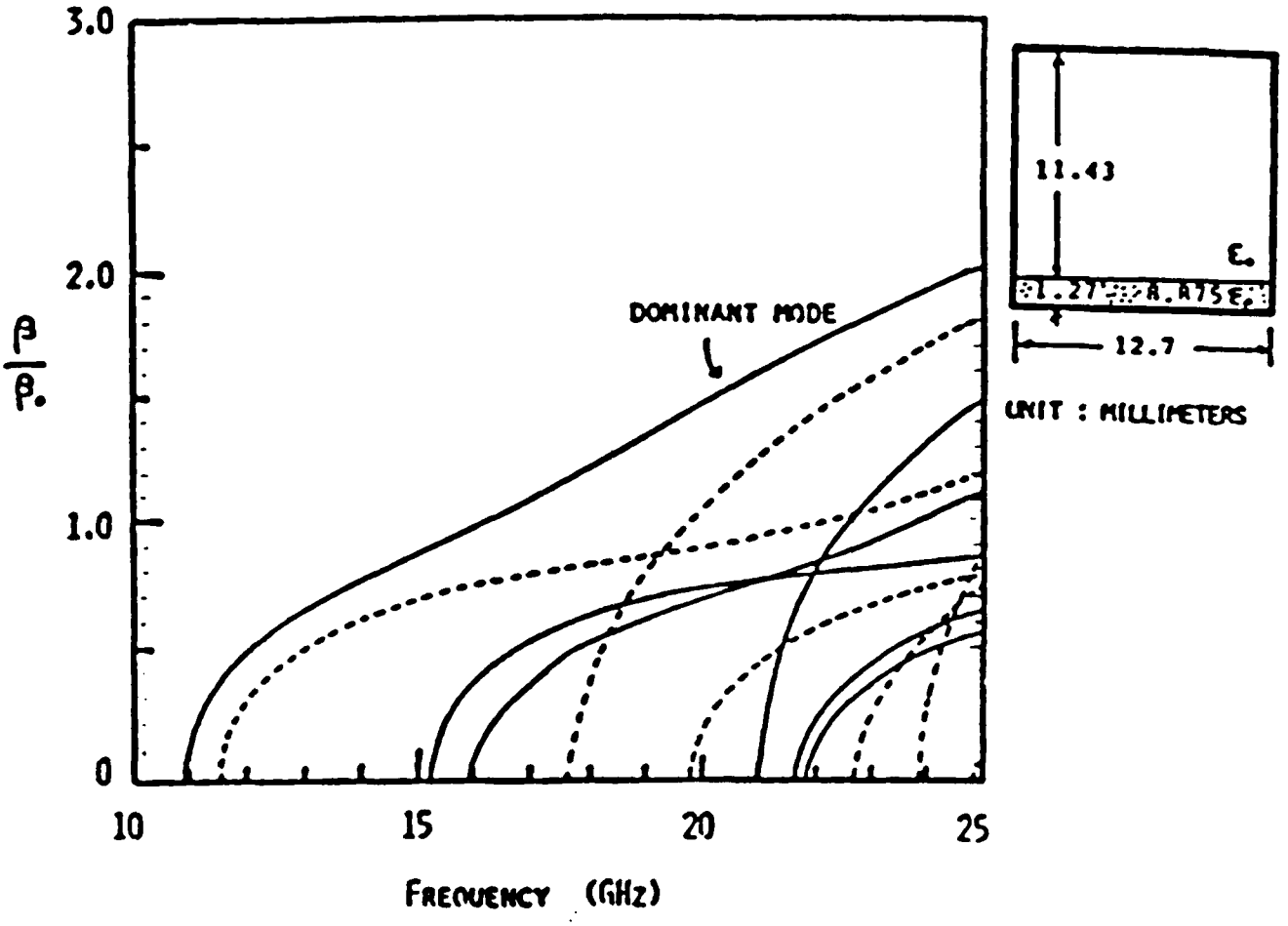


Figure 3: Propagation Constant as a function of frequency (comparison with Yamashita)

I/O FILES AND CODE

.1 screen sample

```
$ p


          THE UNIVERSITY OF MICHIGAN COLLEGE OF ENGINEERING
                         RADIATION LABORATORY
                         ANN ARBOR, MICHIGAN

 ***********************************************************************
 ***********************************************************************

 MMMM     MMMM  MMMM   MMMM  MMMM MMMM  MMMM   MMMM     MMMM   MMMM   MMMM
  MM M   M MM    MM   MM  MM  MM   MM    MM   MM  MM   MM  MM   MM M   MM
  MM  M M  MM    MM  MM       MMMMMMM    MM  MM       MM    MM  MM  M  MM
  MM   M   MM    MM  MM       MM   MM    MM  MM  MMMM MMMMMMMM  MM   M MM
  MM       MM    MM   MM  MM  MM   MM    MM   MM  MM  MM    MM  MM    MMM
 MMMM     MMMM  MMMM   MMMM  MMMM MMMM  MMMM   MMMM  MMMM  MMMM MMMM   MMM

 ***********************************************************************
 ***********************************************************************

 THIS PROGRAM CALCULATES THE PROPAGATION CONSTANT OF THE HYBRID MODES
 EXCITED IN PARTIALLY-FILLED WAVEGUIDES

      T. EMILIE  VAN DEVENTER  -- AND --  LINDA P. B. KATEHI

                                              MARCH 11, 1990 VERSION


Fortran PAUSE
Type return to continue
 Enter name of configuration data file;
yamashita
 ENTER START FREQUENCY (GHz) :
10
 ENTER STOP FREQUENCY (GHz) :
25
 ENTER INCREMENT FREQUENCY (GHz):
1
 Select units:
 [1] inches
 [2] centimeters
2

 Enter # OF LAYERS IN WAVEGUIDE:
2

 Enter permittivity for layer  1
1
 Enter permittivity loss tangent for layer  1
0
 Enter permeability for layer  1
1
 Enter permeability loss tangent for layer  1
0
 Enter height of layer  1
1.143

 Enter permittivity for layer  2
8.875
 Enter permittivity loss tangent for layer  2
0
 Enter permeability for layer  2
1
 Enter permeability loss tangent for layer  2
0
 Enter height of layer  2
.127
 Enter waveguide dimension along x:
1.27
 Enter waveguide dimension along y:
1.27
 Enter maximum # of modes to find
3
 DIMENSIONS ARE TO BE ENTERED IN CENTIMETERS
 ********************************************

Name of geometry configuration file     [NAME] yamashita

the units are CM
cavity dimension along x      [A]   :       1.2700000
cavity dimension along y      [B]   :      1.2700000

 START FREQUENCY TO RUN (GHz)  [FSTR]:  10.00000
 STOP FREQUENCY TO RUN  (GHz)  [FSTP]:  25.00000
 INCREMENT FREQUENCY    (GHz)  [FNCR]:  1.000000

 NUMBER OF DIELECTRIC LAYERS   [NLY] :  2

 RELATIVE ELECTRIC PARAMETERS OF LAYERS:
             dielectric            loss                        permeability
             constant              tangent              real                  imaginary
Layer  1     [EL1R] :    1.000000  [EL1I] :    0.000000 [EL1M] :    1.000000  [M1M]  :       0.000000
Layer  2     [EL2R] :    8.875000  [EL2I] :    0.000000 [EL2M] :    1.000000  [M2M]  :       0.000000

GEOMETRY OF DIELECTRIC LAYERS:          CM
                                    -----------
Layer  1 thickness;      [L1B]  :    1.143000
Layer  2 thickness;      [L2B]  :    0.127000


 PROGRAM PARAMETERS
max. n mode number                       [M]       3
```

CONDUCTOR LOSSES
Include lower ground losses : [GL] F
Include upper ground losses : [UL] F
Conductivity of the walls : [SIG] 0.00000000E+00

Enter Variable Name [**] or <return>

operating frequency	mode excited	kz	propagation constant kz/k0	alpha (Np/m)
11.00000	LSM_ 0 1	31.31126	0.13591	0.00000
12.00000	LSE_ 1 0	79.74963	0.31731	0.00000
12.00000	LSM_ 0 1	120.35992	0.47890	0.00000
13.00000	LSE_ 1 0	135.77371	0.49867	0.00000
13.00000	LSM_ 0 1	174.46198	0.64077	0.00000
14.00000	LSE_ 1 0	177.91298	0.60677	0.00000
14.00000	LSM_ 0 1	223.21684	0.76127	0.00000
15.00000	LSE_ 1 0	214.72264	0.68348	0.00000
15.00000	LSM_ 0 1	272.61243	0.86775	0.00000
16.00000	LSE_ 1 0	248.85355	0.74262	0.00000
16.00000	LSE_ 1 1	27.13669	0.08098	0.00000
16.00000	LSM_ 0 1	326.30457	0.97374	0.00000
16.00000	LSM_ 1 1	126.81892	0.37845	0.00000
16.99999	LSE_ 1 0	281.65442	0.79106	0.00000
16.99999	LSE_ 1 1	134.67567	0.37825	0.00000
16.99999	LSM_ 0 1	386.91238	1.08669	0.00000
16.99999	LSM_ 1 1	188.03099	0.52811	0.00000
17.99999	LSE_ 1 0	314.07829	0.83312	0.00000
17.99999	LSE_ 1 1	193.52902	0.51335	0.00000
17.99999	LSM_ 0 1	455.72186	1.20884	0.00000
17.99999	LSM_ 1 1	233.84915	0.62030	0.00000
17.99999	LSM_ 0 2	155.26567	0.41186	0.00000
18.99999	LSE_ 1 0	347.03436	0.87209	0.00000
18.99999	LSE_ 1 1	243.39511	0.61165	0.00000
18.99999	LSM_ 0 1	532.32367	1.33772	0.00000
18.99999	LSM_ 1 1	272.13550	0.68387	0.00000
18.99999	LSM_ 0 2	315.90094	0.79385	0.00000
19.99999	LSE_ 1 0	381.62134	0.91105	0.00000
19.99999	LSE_ 2 0	87.68021	0.20932	0.00000
19.99999	LSE_ 1 1	290.59106	0.69374	0.00000
19.99999	LSM_ 0 1	614.82782	1.46779	0.00000
19.99999	LSM_ 1 1	305.95987	0.73043	0.00000
19.99999	LSM_ 0 2	440.95157	1.05269	0.00000
20.99999	LSE_ 1 0	419.31534	0.95337	0.00000
20.99999	LSE_ 2 0	200.72998	0.45639	0.00000
20.99999	LSE_ 1 1	338.57596	0.76980	0.00000
20.99999	LSM_ 0 1	700.75470	1.59327	0.00000
20.99999	LSM_ 0 2	554.51080	1.26076	0.00000
20.99999	LSM_ 0 3	39.09147	0.08888	0.00000
21.99999	LSE_ 1 0	461.96701	1.00260	0.00000
21.99999	LSE_ 2 0	272.47369	0.59135	0.00000
21.99999	LSE_ 1 1	390.15616	0.84675	0.00000
21.99999	LSE_ 2 1	114.23763	0.24793	0.00000
21.99999	LSM_ 0 1	787.95502	1.71010	0.00000
21.99999	LSM_ 1 1	365.93430	0.79419	0.00000
21.99999	LSM_ 2 1	71.16162	0.15444	0.00000
21.99999	LSM_ 0 2	661.28510	1.43518	0.00000
21.99999	LSM_ 0 3	362.40820	0.78653	0.00000
22.99999	LSE_ 1 0	511.25302	1.06133	0.00000
22.99999	LSE_ 2 0	328.50262	0.68195	0.00000
22.99999	LSE_ 1 1	447.42371	0.92882	0.00000
22.99999	LSE_ 2 1	216.15340	0.44872	0.00000
22.99999	LSE_ 1 2	128.89127	0.26757	0.00000
22.99999	LSM_ 0 1	874.98932	1.81642	0.00000
22.99999	LSM_ 1 1	393.56696	0.81702	0.00000
22.99999	LSM_ 2 1	180.87822	0.37549	0.00000
22.99999	LSM_ 0 2	762.90967	1.58375	0.00000
22.99999	LSM_ 0 3	525.42639	1.09075	0.00000
23.99999	LSE_ 1 0	567.67902	1.12936	0.00000
23.99999	LSE_ 2 0	374.12955	0.74431	0.00000
23.99999	LSE_ 1 1	510.94788	1.01650	0.00000
23.99999	LSE_ 2 1	280.68002	0.55840	0.00000
23.99999	LSE_ 1 2	278.37512	0.55381	0.00000
23.99999	LSM_ 0 1	961.05621	1.91196	0.00000
23.99999	LSM_ 1 1	420.13651	0.83584	1.78549
23.99999	LSM_ 2 1	244.50510	0.48643	0.00000
23.99999	LSM_ 0 2	860.26398	1.71144	0.00000
23.99999	LSM_ 0 3	658.85938	1.31076	0.00000
24.99999	LSE_ 1 0	630.29608	1.20378	0.00000
24.99999	LSE_ 2 0	412.43860	0.78770	0.00000
24.99999	LSE_ 1 1	579.72522	1.10719	0.00000
24.99999	LSE_ 2 1	330.02109	0.63029	0.00000
24.99999	LSE_ 1 2	390.52069	0.74584	0.00000
24.99999	LSM_ 0 1	1045.78235	1.99730	0.00000
24.99999	LSM_ 1 1	446.21091	0.85220	1.10039
24.99999	LSM_ 2 1	294.31281	0.56210	0.00000
24.99999	LSM_ 0 2	953.98407	1.82198	0.00000
24.99999	LSM_ 1 2	124.01501	0.23685	0.00000
24.99999	LSM_ 0 3	777.25629	1.48445	0.00000

Fortran STOP

.2 FORTRAN code

```
c*************************************************************************
c
c          t.e. van deventer
c          radiation laboratory
c          room 3121, eecs bldg
c          (313) 769-2975
c          department of electrical engineering and computer science
c          the university of michigan
c          ann arbor, mi 48105
c
c*************************************************************************
c
c  program pfw.ftn
c
c  this program is divided into 3 parts,
c  (1) main program
c  (2) subroutines
c  (3) functions
c
c  date  :   o  31 may 1989
c
c bind pfw.bin  muller.bin -b p
c
c  topic :
c    computes the propagation constant of the lse and lsm modes
c    to solve for the transcendental equations of a general
c    n-layers inhomogeneously-filled waveguide
c
c*************************************************************************
c-------------------------
c     define parameters
c-------------------------
      integer q,infer(1),n,nguess,nsig,kn,ms,ligne,n1
      integer itmax,ier,i,l,ifile,lunit,if,mflag
      integer groundup,groundlow,miter
      integer le,m,rdim,mid,r,mdim,index,m1,index1

      real a,b,h(20),sig,ermax,mrmax,unit
      real mu0,eps0,pi,omega,kzr,rs(0:20),rs0(0:20),is(0:20)
      real fop,fopstop,incrfr,fopn
      real epsr(20),mur(20)
      real loss_tan(20),mu_tan(20)
      real old_losstan(20),old_mutan(20)
      real incrkz
      real rwi,iwi,old_incrkz,eps,rkz,fop_1
      real attdbm,old_kzstop,as,old_incrfr
      real absdet,absdet1,amp,differ

      complex sum,sum_1,fsum
      complex kz,kz_1,j,k0,kznoloss
      complex xx(1),zs,kzstart,old_sum,eqn_c
      complex er_c(20),mur_c(20)

      character*50 config_file
      character*2 hmode

      LOGICAL do_gcl,do_ucl
c-------------------------
```

```
c     common blocks
c-------------------------
      common/constant/mu0,eps0,pi,omega,j,k0
      common/mike/er_c,mur_c
      common/geometry/zs,groundup,groundlow
      common/trick/fsum
      common/dimensions/mid
      common/param/mdim,le
      common/geom/a,b
      common/freq/fop,fopstop,incrfr
      common/layers/rdim,h,eps ,mur,loss_tan,mu_tan
      common/operate/do_gcl,do_ucl,sig
      common/mode/m
      common/units/lunit,unit

      external eqn_c
c-------------------------
c      read in input file
c
c   a is the x-direction (or vertical)
c   b is the y-direction (or horizontal)
c   h is the height of the layers
c   fop is the frequency of interest
c   fopstop is the maximum frequency of interest
c   incrfr is the increment of the frequency
c   groundup is a flag ( 1 =  losses in upper wall considered, 0 = no conductor losses )
c   groundlow is a flag ( 1 = losses in lower wall considered, 0 = no conductor losses )
c   incrkz is the increment of the propagation constant kz
c   kzstop is an upper limit to the value of the propagation constant
c   if le is 1, calculate the lse propagation constant
c   if le is 2, calculate the lsm propagation constant
c
c-------------------------
      call logo
      call menuc
c-------------------------
c      open database
c-------------------------
      open(unit=12, file='result',status='unknown')
      call writeout(12)
c-------------------------
      groundup = 0
      groundlow = 0
      if (do_ucl) then
         groundup = 1
      endif
      if (do_gcl) then
         groundlow = 1
      endif

c-------------------------------
c     change values to SI units
c-------------------------------
      fop = fop *1e9
      fopstop = fopstop * 1e9
      incrfr = incrfr * 1e9

      if (lunit .eq. 1) then
```

```
         unit = .0254
      else if (lunit .eq. 2) then
         unit = 1e-2
      endif

      a = a * unit
      b = b * unit
      do 103 r=1,rdim
         h(r) = h(r) * unit
  103 enddo

c----------------------------
c     compute some constants
c----------------------------
      pi = 4. * atan(1.)
      mu0 = 4.e-7 * pi
      eps0 =  1e-9 / (36.*pi)
      j = cmplx(0.,1.)
      mid = nint(rdim / 2.)

      mrmax = 0.0
      ermax = 0.0

      do 101 r=1,rdim
         er_c(r) = epsr(r)*cmplx(1.00,- loss_tan(r))
         mur_c(r) = mur(r)*cmplx(1.00,- mu_tan(r))
         mrmax = amax1(mur(r),mrmax)
         ermax = amax1(epsr(r),ermax)
  101 continue

      ligne = 0
      fop = fop - incrfr
      write (12,*) 'operating        mode                    propag',
     &         'ation constant '
      write (12,*) 'frequency      excited            kz         kz/k0'
     &    ,'     alpha (Np/m)'
      write (12,*) '-------------------------------------------------'
     &    , '--------------------'
      write (*,*) 'operating        mode                    propag',
     &         'ation constant '
      write (*,*) 'frequency      excited            kz         kz/k0'
     &    ,'     alpha (Np/m)'
      write (*,*) '-------------------------------------------------'
     &     ,'--------------------'

c----------------------------
c     frequency loop
c----------------------------
  10  do 5 if=1,1000

         fop = fop + incrfr

         if (fop .gt. fopstop) then
            goto 5000
         endif
         fopn = fop * 1e-9

         omega = 2. * pi * fop
```

```
          k0 = cmplx(omega * sqrt(mu0 * eps0),0.)
          kz = cmplx(.01,0.) * k0
          incrkz = .001 * k0
          miter = 10000 * (sqrt(ermax*mrmax)+.2)

          if (do_ucl .or. do_gcl) then
             amp = sqrt(pi*fop*mu0/sig)
          else
             amp = 0.0
          endif
          zs = cmplx(amp,amp)

          kzstart = kz
          old_incrkz = incrkz

c-----------------------------
c     LSE / LSM loop
c-----------------------------
          do 6 le=1,2

             if (le .eq. 1) then
                hmode = 'E'
             else if (le .eq. 2) then
                hmode = 'M'
             endif

             if (le .eq. 1) then
                 m1 = 0
                 n1 = 1
             else
                 m1 = 1
                 n1 = 0
             endif
c-------------------------
c     m - mode loop
c-------------------------
             do 7 m=m1,mdim

                indexl = n1 - 1
                kz = kzstart

  17            continue

                incrkz = old_incrkz
                kz_1 = cmplx(0.0,0.0)
                sum_1 = cmplx(0.0,0.0)
                mflag = 0

c-------------------------
c  loop to compute the kz root for lossless case
c-------------------------
  12            do 20 ms=1,miter

                   kz = kz + incrkz
                   rkz = real(kz) / real(k0)

                   if (rkz .gt. sqrt(ermax*mrmax)+.2) then
c                      write (*,*) 'beta > quasi-static value'
```

```
                     goto 1000
                  endif

                  sum = cmplx(0.,0.)
                  call hybrid(kz,sum)

c                  write (*,*) 'fop,m,lekz,rkz,sum',fopn,m,le,kz,rkz,sum

c-----------------refinement loop-----------------------------
c
c                 the loop will be used if at least one of the real or imaginary parts
c                 of the sum changed sign since the previous case
c-------------------------
                  if ((real(sum_1) .lt. 0.0 .and. real(sum) .ge. 0.0)
     &            .or. (real(sum_1) .gt. 0.0 .and. real(sum) .le. 0.0)
     &            .or. (aimag(sum).gt.0.0 .and. aimag(sum_1).lt.0.0)
     &            .or. (aimag(sum).lt.0.0 .and. aimag(sum_1).gt.0.0))
     &            then


                     mflag = mflag + 1
                     incrkz = incrkz * 0.1
                     kz = kz_1
c-------------------------
c                    check if the change of sign is due to a singularity in
c                    the function (mflag test) or a root (differ)
c-------------------------
                     absdet = sqrt(real(sum)**2 + aimag(sum)**2)
                     absdet1 = sqrt(real(sum_1)**2 + aimag(sum_1)**2)
                     differ = abs(absdet - absdet1)

                     if (differ .lt. 10.) then
                        goto 13
                     endif
                     if (mflag .gt. 4) then
                        goto 17
                     endif

                     goto 12
                  endif

   27             sum_1 = sum
                  kz_1 = kz

   20          continue
   13          continue
               kznoloss = kz
               index1 = index1 + 1
               rs(index1) = real(kz)
               rs0(index1) = rkz
               is(index1) = 0.0
c-------------------------
c  loop to compute the kz root for lossy case
c--------------initial guess-------------------------------

 40            rwi = real(kz)
               iwi = 0.0
               xx(1) = cmplx(rwi,iwi)
```

```
c--------------muller's method (determinant)----------------

   50           continue

                eps = 1e-3
                nsig = 3
                kn = 0
                nguess = 1
                n = 1
                itmax = 100

                call zanlyt(eqn_c,eps,nsig,kn,nguess,n,xx,itmax,infer
     &          ,ier)

                attdbm = - aimag(xx(1)) * 8.686

                rs(index1) = real(xx(1))
                rs0(index1) = real(xx(1)) / real(k0)
                is(index1) = attdbm

                kz = kznoloss
                goto 17

 1000       continue

            do 153 ii=n1,index1
               i = index1+n1-ii
               ligne = 1
               write (*,655) fopn,hmode,ii,m,rs(i),rs0(i),is(i)
               write (12,655) fopn,hmode,ii,m,rs(i),rs0(i),is(i)
  655          format(f10.5,7x,'LS',A1,'_',i2,i2,6x,f12.5,f12.5,f12.5)
               rs(i) = 0.0
               rs0(i) = 0.0
               is(i) = 0.0
  153       enddo

    7     continue
    6     continue
          if (ligne .eq. 1) then
             write (12,*) '-----------------------------'
             write (*,*) '-----------------------------'
          endif
          ligne = 0
    5 continue

 5000 continue
      stop
      end

cssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss
c                              subroutines
cssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss

       subroutine hybrid(kz,sum)
c------------------------
c  computes the arrays corresponding to lse and lsm common terms
c       le = (1) corresponds to lse modes (electric)
```

```
c       le = (2) corresponds to lsm modes (electric)
c
c     -----cannot have eps or mu equal to zero !!!!---------
c
c-------------------------
      integer m,le,r,rdim,mid,mdim
      integer groundup,groundlow

      real a,b
      real ky
      real h(20),epsr(20),mur(20),loss_tan(20),mu_tan(20)
      real mu0,eps0,pi,omega

      complex k(20),kx(20),kz
      complex j,k0,sum
      complex er_c(20),mur_c(20)
      complex zc(2,20),z0(2,0:21)
      complex twg,zs

      external twg
c-------------------------
      common/constant/mu0,eps0,pi,omega,j,k0
      common/param/mdim,le
      common/mode/m
      common/mike/er_c,mur_c
      common/geometry/zs,groundup,groundlow
      common/geom/a,b
      common/layers/rdim,h,epsr,mur,loss_tan,mu_tan
      common/dimensions/mid

c-------------------------
      ky = m * pi / b

      do 410 r=1,rdim

         k(r) = k0 * csqrt( er_c(r)*mur_c(r) )
         kx(r) = csqrt( k(r)**2 - ky**2 - kz**2)
         zc(1,r) = cmplx(omega * mu0 * mur_c(r)) / kx(r)
         zc(2,r) = kx(r) / cmplx(omega * eps0 * er_c(r))

  410 continue

c-----------------------------------------------------------------
      i = le

      z0(i,0) = zs * groundup
      z0(i,rdim+1) = -zs * groundlow

      do 420 r=1,mid
         z0(i,r) = zc(i,r) *
     &             (z0(i,r-1)+j*zc(i,r)*twg(kx(r)*h(r)))/
     &             (zc(i,r)+j*z0(i,r-1)*twg(kx(r)*h(r)))
 420  continue


      do 430 r=rdim,mid+1,-1
         z0(i,r) = zc(i,r) *
     &             (z0(i,r+1)+j*zc(i,r)*twg(-kx(r)*h(r)))/
```

```
     &             (zc(i,r)+j*z0(i,r+1)*twg(-kx(r)*h(r)))
 430  continue


      sum = z0(i,mid) - z0(i,mid+1)
c      write (*,*) 'kz,i,mid,z0(i,mid),z0(i,mid+1),sum'
c      write (*,*) kz,i,mid,z0(i,mid),z0(i,mid+1),sum
      continue
c-----------------------------------------------------------------
      return
      end
cssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss
      SUBROUTINE MENUC

      INTEGER i,j,i1,i2,n
      INTEGER menu
      INTEGER rdim,mdim,le,lunit

      REAL fop,fopstop,incrfr,sig,unit
      REAL epsr(20),mur(20),loss_tan(20),mu_tan(20)
      REAL a,b
      REAL x
      REAL h(20)

      COMPLEX eps(20)

      LOGICAL do_gcl,do_ucl
      LOGICAL exists

      CHARACTER*80 warning
      CHARACTER*10  check
      CHARACTER*8 ans,un
      CHARACTER*50 config_file
      CHARACTER*50 cfg_file,name_file
      CHARACTER*7 menu_elr(19),menu_eli(19),menu_elm(19)
      CHARACTER*7 menu_layer(19),menu_mu(19)

*******************************************************************************
      common/param/mdim,le
      common/geom/a,b
      common/freq/fop,fopstop,incrfr
      common/layers/rdim,h,epsr,mur,loss_tan,mu_tan
      common/operate/do_gcl,do_ucl,sig
      common/files/config_file
      common/units/lunit,unit

       parameter(PI=3.14159265358979324)

      data menu_elr/'[EL1R]','[EL2R]','[EL3R]','[EL4R]',
     &              '[EL5R]','[EL6R]','[EL7R]','[EL8R]','[EL9R]',
     &              '[EL10R]','[EL11R]','[EL12R]','[EL13R]','[EL14R]',
     &              '[EL15R]','[EL16R]','[EL17R]','[EL18R]','[EL19R]'/
      data menu_eli/'[EL1I]','[EL2I]','[EL3I]','[EL4I]',
     &              '[EL5I]','[EL6I]','[EL7I]','[EL8I]','[EL9I]',
     &              '[EL10I]','[EL11I]','[EL12I]','[EL13I]','[EL14I]',
     &              '[EL15I]','[EL16I]','[EL17I]','[EL18I]','[EL19I]'/
      data menu_elm/'[EL1M]','[EL2M]','[EL3M]','[EL4M]',
     &              '[EL5M]','[EL6M]','[EL7M]','[EL8M]','[EL9M]',
```

```
     &              '[EL10M]','[EL11M]','[EL12M]','[EL13M]','[EL14M]',
     &              '[EL15M]','[EL16M]','[EL17M]','[EL18M]','[EL19M]'/
      data menu_mu/'[M1M]','[M2M]','[M3M]','[M4M]',
     &              '[M5M]','[M6M]','[M7M]','[M8M]','[M9M]',
     &              '[M10M]','[M11M]','[M12M]','[M13M]','[M14M]',
     &              '[M15M]','[M16M]','[M17M]','[M18M]','[M19M]'/
      data menu_layer/'[L1B]','[L2B]','[L3B]','[L4B]',
     &              '[L5B]','[L6B]','[L7B]','[L8B]','[L9B]',
     &              '[L10B]','[L11B]','[L12B]','[L13B]','[L14B]',
     &              '[L15B]','[L16B]','[L17B]','[L18B]','[L19B]'/

*******************************************************************************
*      DEFINE WRITES A FILE CONTAINING ALL OF THE PARAMETERS NEEDED FOR
*      THE ANALYSIS OF THE INHOMOGENEOUSLY-FILLED WAVEGUIDE PROBLEM.
*      THE VARIABLES AND PARAMETERS ARE NAMED AS FOLLOWS:
*
*      a                      -  dimension of cavity along x axis
*      b                      -  dimension of cavity along y axis
*      fop                    -  frequency of analysis in GHz
*      fopstop                -  maximum frequency
*      incrfr                 -  frequency increment
*      rdim                   -  number of layers in cavity substrate
*******************************************************************************
C     THE NEXT SEGMENT DETERMINES WHETHER AN EXISTING CONFIGURATION FILE IS
C     TO BE USED
*******************************************************************************
  100 write(*,*) 'Enter name of configuration data file; '
      read(*,'(A50)') config_file              {geometric configuration file name
      cfg_file=config_file
      inquire(file=cfg_file,exist=exists) {check if format file already exist
      call trim(cfg_file,50,i1,i2)
      if (exists ) then
          warning = '   WARNING: File '//cfg_file(i1:i2)//
     &              ' already exists.  Use existing file?'//
     &              ' ( <return> = yes )'
          write (6,*) warning
          read (5,'(a1)') ans
          if ((ans.ne.'y').and.(ans.ne.'Y').and.(ans.ne.'')) goto 100
          goto 200
      endif

*******************************************************************************
C     NEXT SECTION OF CODE IS ONLY REACHED IF NO CONFIGURATION FILE ALREADY
C     EXISTS AND WILL COLLECT ALL OF THE NECESSARY INPUT AND GENERATE A NEW
C     CONFIGURATION
*******************************************************************************

  949 write(*,*) 'ENTER START FREQUENCY (GHz) : '
      read(*,*,err=949) fop
  950 write(*,*) 'ENTER STOP FREQUENCY (GHz) : '
      read(*,*,err=949) fopstop
  951 write(*,*) 'ENTER INCREMENT FREQUENCY (GHz): '
      read(*,*,err=949) incrfr

      write(*,*) 'Select units: '
      write(*,*) '[1] inches '
      write(*,*) '[2] centimeters '
      read(*,*) lunit
```

```
      write(*,*)
      write(*,*) 'Enter # OF LAYERS IN WAVEGUIDE: '
      read(*,*) rdim

      x = 0.0
      do i = 1, rdim

         write(*,*)
         write(*,*)
     &      'Enter permittivity for layer ',i
         read(*,*) epsr(i)
         if (epsr(i) .eq. 0.0) epsr(i) = 1.0
         write(*,*)
     &      'Enter permittivity loss tangent for layer ',i
         read(*,*) loss_tan(i)
         write(*,*)
     &      'Enter permeability for layer ',i
         read(*,*) mur(i)
         if (mur(i) .eq. 0.0)  mur(i) = 1.0
         write(*,*)
     &      'Enter permeability loss tangent for layer ',i
         read(*,*) mu_tan(i)


         write(*,*) 'Enter height of layer ',i
         read(*,*) h(i)

         x = x + h(i)
      enddo

*****************************************************************************
*     WAVEGUIDE GEOMETRY
*****************************************************************************
      write(*,*) 'Enter waveguide dimension along x: '
      read(*,*) a

      write(*,*) 'Enter waveguide dimension along y: '
      read(*,*) b

      if (x .ne. a) then
         goto 499
      endif

*****************************************************************************
*     NUMERICAL PARAMETERS
*****************************************************************************

      write(*,*) 'Enter maximum # of modes to find'
      read(*,*) mdim

      do_gcl = .false.
      do_ucl = .false.
      le = 2
      sig = 0.0

      goto 400
```

```
C*******************************************************************************
*     THE NEXT SEGMENT READS DATA FROM EXISTING FILE
C*******************************************************************************

  200 open(10,file=cfg_file)

      read (10,*)
      read (10,650) config_file
      read (10,*)
      read (10,*) mdim
      read (10,*)
      read (10,*) lunit
      read (10,*)
      read (10,*) a,b
      read (10,*)
      read (10,*) fop,fopstop,incrfr
      read (10,*)
      read (10,*) rdim
      read (10,*)
      do i = 1,rdim
         read(10,*) h(i),epsr(i),mur(i),loss_tan(i),mu_tan(i)
      enddo
      read (10,*)
      read (10,*) do_gcl
      read (10,*) do_ucl
      read (10,*)
      read (10,*) sig
      read (10,*)
      read (10,*)
  650 format(A50)

      close(10)

*******************************************************************************
C   THE NEXT PART IS THE MAIN PART OF THE MENU SYSTEM
*******************************************************************************

*******************************************************************************
*               WRITE VARIABLE LIST ON SCREEN
*******************************************************************************
  400 continue

      if (lunit.eq.2) then
         write(*,*) 'DIMENSIONS ARE TO BE ENTERED IN CENTIMETERS '
         un = 'CM'
      else if (lunit.eq.1) then
         write(*,*) 'DIMENSIONS ARE TO BE ENTERED IN INCHES '
         un = 'INCH'
      endif


         write(*,*)
         write(*,*) '************************************************'
         write(*,*)
         write(*,705) config_file
         write(*,*)
         write(*,704) un
         write(*,710) a
```

```
      write(*,715) b
      write(*,*)

  704 format('the units are ',A7)
  705 format('Name of geometry configuration file     [NAME] ', A50)
  710 format('cavity dimension along x      [A]   : ',
     & f14.7)
  715 format('cavity dimension along y      [B]   :',
     & f14.7)

      write(*,*) 'START FREQUENCY TO RUN (GHz)  [FSTR]: ' ,fop
      write(*,*) 'STOP FREQUENCY TO RUN  (GHz)  [FSTP]: ' ,fopstop
      write(*,*) 'INCREMENT FREQUENCY    (GHz)  [FNCR]: ' ,incrfr
      write(*,*)
c 716 format('frequency',i3,' in GHz',3x,A7,2x,f14.7)

      write(*,*)
      write(*,*) 'NUMBER OF DIELECTRIC LAYERS   [NLY] : ',rdim
      if (rdim.gt.1) then
         write(*,*)
         write(*,*) 'RELATIVE ELECTRIC PARAMETERS OF LAYERS:'
         write(*,*) '              dielectric    ',
     &      '        loss                          permeability'
         write(*,*) '              constant      ',
     &      '        tangent                 real              ',
     &      ' imaginary'
         do i = 1,rdim
            write(*,720) i,menu_elr(i),epsr(i),menu_eli(i),
     &           loss_tan(i),menu_elm(i),mur(i),menu_mu(i),
     &           mu_tan(i)
         enddo

         write(*,*)
         write(*,718) un
         write(*,*) '                                    ',
     &       ' -----------'
         do i = 1,rdim
            write(*,725) i,menu_layer(i),h(i)
         enddo

         write(*,*)
      endif
  718 format('GEOMETRY OF DIELECTRIC LAYERS:      ',2x,A7)
  720 format('Layer',i3,5x,A7,':',f12.6,2x,A7,':',f12.6,
     & 2x,A7,':',f12.6,2x,A7,':',f12.6)
  725 format('Layer',i3,' thickness;      ',A7,':',f12.6)

      write(*,*)
      write(*,*) 'PROGRAM PARAMETERS '
      write(*,777) mdim
      write(*,*)
      write(*,*) 'CONDUCTOR LOSSES '
      write(*,744) do_gcl
      write(*,745) do_ucl
      write(*,746) sig

  777 format('max. n mode number                        [M]  ', i5)
  744 format('Include lower ground losses     :',8x,'[GL]  ',11,5x)
```

```
  745 format('Include upper ground losses      :',8x,'[UL]  ',11,5x)
  746 format('Conductivity of the walls        :',8x,'[SIG] ',e15.8)

c      goto 9999
*******************************************************************************
*******************************************************************************
*******************************************************************************
C     THE NEXT PART READS THE RESPONSES TO THE MENU PROMPT
C     DETECTS WHICH VARIABLE IS TO BE CHANGED
C     PROMPTS FOR THE CHANGE AND MAKES IT
C     AND UPDATES THE MENU
*******************************************************************************


      write(*,*)
      write(*,*) 'Enter Variable Name [**] or <return>'

      read(*,'(A4)',err=499) check

********  ALTER VARIABLES AS DESIRED

      if (check.eq.'') goto 600

      if (    (check.eq.'STOP') .or. (check.eq.'stop')
     &    .or.(check.eq.'QUIT') .or. (check.eq.'quit')
     &    .or.(check.eq.'EXIT') .or. (check.eq.'exit')
     &    .or.(check.eq.'ABORT').or. (check.eq.'abort') ) stop

      if ((check.eq.'name').or.(check.eq.'NAME')) then
  501   write(*,*) 'ENTER NEW CONFIGURATION FILE NAME '
        read(*,'(a50)',err=499) name_file
        call trim(name_file,50,i1,i2)
         inquire(file=name_file(i1:i2),exist=exists)
         if(exists) then
          warning = '   WARNING: File '//name_file(i1:i2)//
     &              ' already exists. Input existing file?'//
     &              ' ( <return> = yes )'
          write (6,*) warning
          read (5,'(a1)') ans
          if ((ans.ne.'y').and.(ans.ne.'Y').and.(ans.ne.'')) goto 501
          config_file=name_file
          cfg_file=name_file
          goto 200
        else
          config_file=name_file
          cfg_file=name_file
          goto 400
        endif
      endif

      if ((check.eq.'a').or.(check.eq.'A')) then
        write(*,*) 'ENTER A '
        read(*,*,err=499) a
        goto 400
      endif

      if ((check.eq.'b').or.(check.eq.'B')) then
        write(*,*) 'ENTER B '
```

```
        read(*,*,err=499) b
        goto 40
      endif

      if ((check.eq.'fstr').or.(check.eq.'FSTR')) then
        write(*,*) 'ENTER START FREQUENCY (in GHz)'
        read(*,*,err=499) fop
        goto 400
      endif

      if ((check.eq.'fstp').or.(check.eq.'FSTP')) then
        write(*,*) 'ENTER STOP FREQUENCY (in GHz)'
        read(*,*,err=499) fopstop
        goto 400
      endif

      if ((check.eq.'fncr').or.(check.eq.'FNCR')) then
        write(*,*) 'ENTER FREQUENCY INCREMENT (in GHz)'
        read(*,*,err=499) incrfr
        goto 400
      endif

      if ((check.eq.'nly').or.(check.eq.'NLY')) then
        write(*,*) 'ENTER # OF DIELECTRIC LAYERS'
        read(*,*,err=499) rdim
        goto 400
      endif

      do i = 1, rdim
         if ((check(1:1).eq.'e').or.(check(1:1).eq.'E')) then
            if ((check(2:2).eq.'l').or.(check(2:2).eq.'L')) then
               if ((check(4:4).eq.'r').or.(check(4:4).eq.'R')) then
                  if (check(3:3).eq.menu_elr(i)(4:4)) then
                     write(*,*) 'ENTER ',menu_elr(i)
                     read(*,*,err=499) epsr(i)
                     eps(i)= epsr(i) * cmplx(1.,loss_tan(i))
                     goto 400
                  endif
               else if ((check(4:4).eq.'i').or.(check(4:4).eq.'I')) then
                  if (check(3:3).eq.menu_eli(i)(4:4)) then
                     write(*,*) 'ENTER ',menu_eli(i)
                     read(*,*,err=499) loss_tan(i)
                     eps(i) = epsr(i) * cmplx(1.,loss_tan(i))
                     goto 400
                  endif
               else if ((check(4:4).eq.'m').or.(check(4:4).eq.'M')) then
                  if (check(3:3).eq.menu_elm(i)(4:4)) then
                     write(*,*) 'ENTER ',menu_elm(i)
                     read(*,*,err=499) mur(i)
                     goto 400
                  endif
               else if ((check(5:5).eq.'r').or.(check(5:5).eq.'R')) then
                  if (check(4:4).eq.menu_elr(i)(5:5)) then
                     write(*,*) 'ENTER ',menu_elr(i)
                     read(*,*,err=499) epsr(i)
                     eps(i)= epsr(i) * cmplx(1.,loss_tan(i))
                     goto 400
                  endif
```

```
            else if ((check(5:5).eq.'i').or.(check(5:5).eq.'I')) then
               if (check(4:4).eq.menu_eli(i)(5:5)) then
                  write(*,*) 'ENTER ',menu_eli(i)
                  read(*,*,err=499) loss_tan(i)
                  eps(i) = epsr(i) * cmplx(1.,loss_tan(i))
                  goto 400
               endif
            else if ((check(5:5).eq.'m').or.(check(5:5).eq.'M')) then
               if (check(4:4).eq.menu_elm(i)(5:5)) then
                  write(*,*) 'ENTER ',menu_elm(i)
                  read(*,*,err=499) mur(i)
                  goto 400
               endif
            endif
         endif
      endif
      if ( (check(1:1).eq.'l').or.(check(1:1).eq.'L') ) then
         if ( (check(3:3).eq.'b').or.(check(3:3).eq.'B') ) then
            if ( check(2:2).eq.menu_layer(i)(3:3) ) then
               write(*,*) 'ENTER ',menu_layer(i)
               read(*,*,err=499) h(i)
               goto 400
            endif
         else if ((check(4:4).eq.'b').or.(check(4:4).eq.'B')) then
            if ( check(3:3).eq.menu_layer(i)(4:4) ) then
               write(*,*) 'ENTER ',menu_layer(i)
               read(*,*,err=499) h(i)
               goto 400
            endif
         endif
      endif
      if ( (check(1:1).eq.'m').or.(check(1:1).eq.'M') ) then
         if ( (check(3:3).eq.'m').or.(check(3:3).eq.'M') ) then
            if ( check(2:2).eq.menu_mu(i)(3:3) ) then
               write(*,*) 'ENTER ',menu_mu(i)
               read(*,*,err=499) mu_tan(i)
               goto 400
            endif
         else if ((check(4:4).eq.'m').or.(check(4:4).eq.'M')) then
            if ( check(3:3).eq.menu_mu(i)(4:4) ) then
               write(*,*) 'ENTER ',menu_mu(i)
               read(*,*,err=499) mu_tan(i)
               goto 400
            endif
         endif
      endif
   enddo


   if ((check.eq.'m').or.(check.eq.'M')) then
     write(*,*) 'ENTER M'
     read(*,*,err=499) mdim
     goto 400
   endif

   if ((check.eq.'gl').or.(check.eq.'GL')) then
     do_gcl = .not.do_gcl
     goto 400
```

```
      endif

      if ((check.eq.'ul').or.(check.eq.'UL')) then
        do_ucl = .not.do_ucl
        goto 400
      endif

      if ((check.eq.'sig').or.(check.eq.'SIG')) then
        write(*,*) 'ENTER CONDUCTIVITY OF THE WALLS (S/m)'
        read(*,*,err=499) sig
        goto 400
      endif

  499 write(*,*)
      write(*,*) '*********************************************'
      write(*,*) '      Entry format error -- try again        '
      write(*,*) '*********************************************'
      write(*,*)
      goto 400


*******************************************************************************
C     THE NEXT PART IS THE LAUNCHING POINT FOR THE REST OF THE PROGRAM
C     IT IS ONLY REACHED WHEN FINISHED WITH THE MENU PART
*******************************************************************************

*******************************************************************************

  600 continue

*******************************************************************************
********  WRITE VARIABLES TO NEW FILE OR OLD FILE
  500 call trim(cfg_file,50,i1,i2)
      inquire(file=cfg_file,exist=exists)
      if(exists) then
        write(*,*) 'File already exists --- ',
     &             'overwrites not allowed.'
        write(*,*) 'Rename [R] or <return> (-- changes not saved)'
        read(*,'(A8)') ans
        if (ans.ne.'') then
           if (ans.ne.'override') then
              goto 501
           else
              goto 599
           endif
        endif
      else
  599   open(10,file=cfg_file,status='write')
        CALL WRITEOUT(10)
        close(10)
      endif
*******************************************************************************

 9999 return
      end

*******************************************************************************
*******************************************************************************
```

```
*******************************************************************************
      subroutine trim(char_str,length_spec,i1,i2)
c*****************************************************************c
c                                                                 c
c   finds positions of the first and last non-blank characters    c
c   in a string                                                   c
c                                                                 c
c*****************************************************************c
      integer length_spec,i1,i2,i
      character*(*) char_str
      do 10 i=1,length_spec
        if (char_str(i:i).ne.' ') then
          i1=i
          goto 15
        endif
   10 continue
   15 do 20 i=length_spec,1,-1
        if (char_str(i:i).ne.' ') then
          i2=i
          goto 25
        endif
   20 continue
   25 return
      end



*******************************************************************************
*******************************************************************************
*******************************************************************************
*     SUBROUTINE WRITEOUT WRITES ALL OF THE PARAMETERS FOR THE PROBLEM IN
*     THE FILE OPENED AS file = ifile IN THE CALLING PROGRAM.
*
*     THE NORMALIZED PARAMETERS (a,b,c,....) ARE PASSED IN THE COMMON STATEMENT
*     AND WHEN WRITEOUT IS CALLED ARE ASSUMED NORMALIZED TO FREE SPACE WAVELENGTH
*     WRITE OUT RENORMALIZES TO WAVELENGTHS IN THE HOST MEDIA
*     BEFORE WRITING OUT TO THE DATA FILE.
*******************************************************************************

      SUBROUTINE WRITEOUT(ifile)

      integer mdim,le,rdim,lunit
      integer ifile,i,j,n

      real a,b
      real fop,fopstop,incrfr,unit,sig
      real h(20),epsr(20),mur(20),loss_tan(20),mu_tan(20)

      logical do_gcl,do_ucl

      character*50 config_file

      common/param/mdim,le
      common/geom/a,b
      common/freq/fop,fopstop,incrfr
      common/layers/rdim,h,epsr,mur,loss_tan,mu_tan
      common/operate/do_gcl,do_ucl,sig
```

```
      common/files/config_file
      common/units/lunit,unit

      write (ifile,*) '***  input data file :'
      write (ifile,*) config_file
      write (ifile,*) '***  maximum mode number :'
      write (ifile,*) mdim
      write (ifile,*) '***  units (1=inch, 2=cm) :'
      write (ifile,*) lunit
      write (ifile,*) '***  x and y dimensions :'
      write (ifile,*) a,b
      write (ifile,*) '***  start,stop and incrmt frequencies in GHz :'
      write (ifile,*) fop,fopstop,incrfr
      write (ifile,*) '***  number of layers :'
      write (ifile,*) rdim
      write (ifile,*) '***  height, epsr, mur, loss_tan(epsr),'
     &      ,' loss_tan(mur) :'
      do i = 1,rdim
         write(ifile,*) h(i),epsr(i),mur(i),loss_tan(i),mu_tan(i)
      enddo
      write (ifile,*) '***  losses below and above walls :'
      write (ifile,*) do_gcl
      write (ifile,*) do_ucl
      write (ifile,*) '***  conductivity of walls :'
      write (ifile,*) sig
      write (ifile,*)
      write (ifile,*)

      return
      end
C-----------------------------------------------------------------------
      SUBROUTINE LOGO
            THE UNIVERSITY OF MICHIGAN COLLEGE OF ENGINEERING
                        RADIATION LABORATORY
                        ANN ARBOR, MICHIGAN

 ************************************************************************
 ************************************************************************

 MMMM      MMMM   MMMM    MMMM   MMMM MMMM   MMMM    MMMM       MMMM     MMMM    MMMM
  MM M    M MM     MM    MM  MM   MM    MM     MM    MM  MM     MM  MM     MM M    MM
  MM  M M  MM     MM   MM        MMMMMMMM     MM   MM          MM    MM    MM  M   MM
  MM   M   MM     MM   MM        MM    MM     MM   MM  MMMM  MMMMMMMMM    MM    M MM
  MM       MM     MM    MM  MM   MM    MM     MM   MM  MM    MM     MM    MM      MMM
 MMMM      MMMM   MMMM    MMMM   MMMM MMMM   MMMM    MMMM   MMMM   MMMM  MMMM     MMM

 ************************************************************************
 ************************************************************************


                                                    MARCH 11, 1990 VERSION


      WRITE(*,*) ' '
      WRITE(*,*) ' '
      WRITE(*,*) ' '
      WRITE(*,*) ' '
      WRITE(*,*) ' '
```

```
      WRITE(*,*) ' '
      WRITE(*,*) '          THE UNIVERSITY OF MICHIGAN COLLEGE OF ENGI',
     &           'NEERING'
      WRITE(*,*) '                          RADIATION LABORATORY'
      WRITE(*,*) '                          ANN ARBOR, MICHIGAN'
      WRITE(*,*) ' '
      WRITE(*,*) '*****************************************************',
     &           '************************'
      WRITE(*,*) '*****************************************************',
     &           '************************'
      WRITE(*,*) ' '
      WRITE(*,*) 'MMMM      MMMM  MMMM   MMMM   MMMM MMMM  MMMM    MMMM  ',
     &           '    MMMM    MMMM    MMMM'
      WRITE(*,*) ' MM M    M MM     MM    MM   MM   MM    MM     MM     MM   MM ',
     &           '    MM   MM     MM M    MM'
      WRITE(*,*) ' MM  M M  MM      MM   MM         MMMMMMM     MM   MM       ',
     &           '   MM    MM    MM  M   MM'
      WRITE(*,*) ' MM   M   MM      MM   MM         MM    MM    MM   MM  MMMM',
     &           '   MMMMMMMM    MM    M MM '
      WRITE(*,*) ' MM         MM      MM    MM   MM   MM    MM     MM     MM   MM ',
     &           '   MM     MM     MM     MMM'
      WRITE(*,*) 'MMMM      MMMM  MMMM   MMMM   MMMM MMMM  MMMM    MMMM  ',
     &           ' MMMM  MMMM MMMM    MMM'
      WRITE(*,*) ' '
      WRITE(*,*) '*****************************************************',
     &           '************************'
      WRITE(*,*) '*****************************************************',
     &           '************************'
      WRITE(*,*) ' '
      WRITE(*,*) 'THIS PROGRAM CALCULATES THE PROPAGATION CONSTANT ',
     &           'OF THE HYBRID MODES'
      WRITE(*,*) 'EXCITED IN PARTIALLY-FILLED WAVEGUIDES'
      WRITE(*,*) ' '
      WRITE(*,*) '     T. EMILIE  VAN DEVENTER  -- AND --  LINDA P. B.',
     &           ' KATEHI'
      WRITE(*,*) ' '
      WRITE(*,*) '                                                    ',
     &           '  MARCH 11, 1990 VERSION'
      WRITE(*,*) ' '
      WRITE(*,*) ' '
      WRITE(*,*) ' '
      WRITE(*,*) ' '
      PAUSE

      RETURN
      END

cffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
c                              functions
cffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
      complex function eqn_c(kz)
c
      complex sum,fsum,kz
      common/trick/fsum

      sum = cmplx(0.,0.)
      call hybrid(kz,sum)
      eqn_c = sum
```

```
      fsum=sum

      return
      end

cffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
      complex function twg(arg)
      complex arg,j
c-----------------
      j = cmplx(0.0,1.0)
      if(aimag(arg) .gt. 45.0) then
         twg = j
      else if (aimag(arg) .lt. -45.0) then
         twg = -j
      else
         twg = csin(arg) / ccos(arg)
      endif
      return
      end
cffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
```

References

[1] R.F. Harrington, "Time-Harmonic Electromagnetic Fields", McGraw-Hill, 1961.

[2] IMSL User's Manual, IMSL Library, Problem-Solving Software System For Mathematical and Statistical FORTRAN Programming, Edition 9.2, Revised November 1984, IMSL LIB-0009.

[3] E. Yamashita, "Analysis of Microstrip-Like Transmission Lines by Nonuniform Discretization of Integral Equations," *IEEE Trans. Microwave Theory Tech.*, vol. MTT-24, No. 4, pp.195-200, April 1976.